CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Partial Decomposition Strategies for Two-Stage Stochastic Integer Programs

Teodor Gabriel Crainic
Mike Hewitt
Walter Rei

March 2014

CIRRELT-2014-13

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

UNIVERSITÉ LAVAL · McGill · Concordia · ÉTS · UQÀM · HEC MONTRÉAL · POLYTECHNIQUE MONTRÉAL · Université de Montréal

Partial Decomposition Strategies for Two-Stage Stochastic Integer Programs

Teodor Gabriel Crainic[1], Mike Hewitt[2], Walter Rei[1,*]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

[2] Department of Information Systems & Operations Management, Loyola University Chicago, 820 N. Michigan Ave, Chicago, IL, USA 60611

Abstract. We propose the concept of partial Benders decomposition, based on the idea of retaining a subset of scenario subproblems in the master formulation and develop a theory to support it that illustrates how it may be applied to any stochastic integer program with continuous recourse. Such programs are used to model many practical applications such as the one considered in this paper, network design. They are also useful for solving problems with integer recourse as many solution methods for such problems also solve one of its linear relaxations. With an extensive computational study, we have shown the significant advantages of using a partial decomposition, greatly reducing the number of optimality and feasibility cuts generated when solving a stochastic program with a Benders-based algorithm. We also show that how the partial decomposition is performed has a significant impact and point to the most performant strategy.

Keywords: Partial decomposition, Benders method, mixed-integer stochastic programming, network design.

Acknowledgements. Partial funding for this project has been provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Fonds de Recherche du Québec - Nature et technologies (FQRNT). This support is gratefully acknowledged.

Results and views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect those of CIRRELT.

Les résultats et opinions contenus dans cette publication ne reflètent pas nécessairement la position du CIRRELT et n'engagent pas sa responsabilité.

---

* Corresponding author: Walter.Rei@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2014

© Crainic, Hewitt, Rei and CIRRELT, 2014

1 Introduction

Since its introduction in 1962, Benders decomposition [2] has become one of the most used exact solution approaches for large-scale optimization problems. It has been shown to be an efficient solution methodology for a variety of applications, such as network design [8] and location [13, 7], and has been successfully used in specialized optimization fields, as evidenced by such seminal papers as [17, 37] for non-linear and stochastic programming, respectively.

The Benders algorithm is now an essential methodology in the context of stochastic programming, readily applied to problems addressed in the field [3]. Stochastic programming deals with optimization problems where a subset of parameters involve a level of uncertainty (i.e., stochastic parameters). Decisions in a stochastic program are defined in stages according to when the stochastic parameters become known. Therefore, one distinguishes the decisions that need to be made before any information is known (i.e., the first stage decisions) from the decisions that are taken once the informational flow begins (i.e., the second stage decisions and onward). First stage decisions are sometimes referred to as the *a priori* decisions, while those made in subsequent stages are called *recourse* decisions [3]. The pursued objective can then be defined as finding an a priori solution, which minimizes its associated cost plus a probabilistic measure of the recourse cost that it entails, e.g., the expected cost, a value at risk, or an expected shortfall cost [30].

When it was originally developed in [37], the Benders strategy applied to stochastic programs was coined the L-Shaped algorithm. It enables such programs to be decomposed according to the realizations of random events that set the values of the stochastic parameters included in the model. A finite set of representative scenarios is usually used to approximate the possible outcomes for the values of the stochastic parameters. Using such a set, the stochastic program can be formulated in an extensive form by duplicating the second stage decisions for each scenario [3]. Given that the large-scale nature of such models is due, in a large proportion, to the number of scenarios used to properly represent uncertainty, Benders decomposition greatly simplifies the solution of these problems. However, this strategy also comes with important drawbacks that need to be addressed to produce an overall efficient solution procedure. In this paper, we focus on two stage stochastic integer programs, which define an extremely challenging family of optimization problems to solve. We show that the L-Shaped algorithm, when applied to solve this type of problem, can be significantly enhanced by modifying the decomposition strategy that is traditionally applied.

As presented in [15, 16], Benders decomposition relies on the application of the following three steps: *Step 1 Projection* - the model is first projected onto the subspace defined by a set of variables considered to be complicating (i.e., the first stage decisions when the L-Shaped algorithm is applied in the present context); *Step 2 Dualization* - the projected term is then dualized, which produces an equivalent model where the projected term is expressed as a set of valid inequalities (or cuts) that define, for the

complicating variables, their feasibility requirements (feasibility cuts) and their projected costs (optimality cuts); *Step 3 Relaxation* - the equivalent model is finally solved by applying a relaxation strategy where a master problem and subproblem (or scenario subproblems in the case considered) are iteratively solved to respectively guide the search process and generate violated cuts.

The main drawback when applying the L-Shaped algorithm is that the initial relaxation of the equivalent model produces a considerably weaker formulation for the obtained master problem. The cuts included in the equivalent model provide the formulation of the second stage of the stochastic problem. Once they are relaxed, the master problem looses all relevant information concerning the recourse decisions, both in terms of the projected costs and feasibility of the scenario subproblems. When applying the original L-Shaped method, these cuts are reintroduced iteratively by solving each time the relaxed master problem. Overall, this leads to various computational problems such as instability issues with respect to the cuts that are generated (especially at the beginning of the solution process); an erratic progression of the bounds generated by the algorithm; and an overall slow convergence of the procedure (the relaxed master problem is solved each time a cut is generated).

We define a novel decomposition concept to strengthen the master problems solved by the algorithm and thus overcome these challenges. This decomposition is based on the idea of retaining a subset of scenario subproblems in the master formulation. Specifically, we propose to project the original two stage stochastic integer program onto a larger subspace, one that includes a subset of the second stage scenario variables. We refer to this approach as *partial Benders decomposition*. By doing so, the formulation of the master problem is automatically improved. In effect, a smaller part of the original problem is relaxed at the beginning of the solution process. However, this is done at the expense of harder master problems to solve. Ultimately, we show that this approach can greatly improve the L-Shaped algorithm, the level of these improvements depending on how the partial decomposition is applied.

There are two major issues to address when considering the application of partial decomposition. One must determine whether the decomposition can actually enhance the Benders algorithm or not. If so, one must specify how to select the scenario subproblems to retain. We resolve these issues by proposing a series of strategies implementing partial decomposition that aim to reduce the number of cuts (optimality and feasibility) added by the L-Shaped algorithm in order to converge. Two general types of strategies are developed: representation and covering. *Representation* strategies choose scenario subproblems to retain that serve as representatives of the ones not retained, while *covering* strategies retain scenario subproblems that exhibit specific characteristics with respect to the ones not kept. In all cases, these strategies are based on the general principle of identifying scenario subproblems to keep in the master that best fit the desirable criteria that we define to reduce the number of cuts.

Extensive numerical experiments were conducted on a set of stochastic network design problems to evaluate the strategies proposed. The results obtained show the

significant advantages of partial decomposition. In particular, partial decomposition reduces by an order of magnitude the number of cuts generated by the Benders algorithm compared to a standard implementation of the solution approach. Furthermore, the results illustrate that specific strategies are more effective depending on the characteristics of the problem.

The rest of the paper is divided as follows. In Section 2, we recall how Benders decomposition is applied to two stage stochastic integer programs and provide a literature review of the different methods that have been proposed over the years to improve the algorithm. section 3 is dedicated to the presentation of partial decomposition, starting with general principles and motivation, and then defining the specific representation and covering strategies proposed. Section 4 first details the experimental setting, the problem considered, the instances used, and the different implementations of the L-Shaped algorithm tested, and then presents the analysis of the numerical results. We provide concluding remarks in Section 5.

2 A review of Benders decomposition

We start by recalling how Benders decomposition is applied to the problems considered (Section 2.1) and recall the drawbacks associated to the method. We then review the different strategies that have been developed to improve the algorithm (Section 2.2), covering both the strategies that were proposed to accelerate the algorithm in the general context (i.e., to solve mixed integer programs) and those that were specifically designed to enhance the L-Shaped procedure. This enables us to clearly state the novelty of the contributions made in the present paper.

2.1 Benders decomposition for two stage stochastic integer programs

Let y define a set of first stage decision variables that must take on integer values and satisfy the constraint set $Ay = b$. Next, we assume there are S scenarios, with each scenario $s \in S$ having a probability p_s of occuring. We associate with each scenario $s \in S$ a set of second stage decision variables x^s that, together with the first stage decisions y, must satisfy the constraints $B^s y + Dx^s = d^s$. With objective coefficients f associated with the y variables, and c the objective coefficients associated with the x^s variables for each scenario s, we have the optimization problem P:

$$\text{minimize } f^\top y + \sum_{s \in S} p_s c^\top x^s \tag{1}$$

subject to

$$Ay = b, \tag{2}$$

$$B^s y + Dx^s = d^s, \quad \forall s \in S, \tag{3}$$

$$y \in \mathscr{Z}, x^s \geq 0 \ \ s \in S. \tag{4}$$

Recourse decisions x^s, $\forall s \in S$, could be restricted to take on integer values. In this paper, we address the case of continuous recourse decisions, as it allows us to apply standard duality theory when presenting the Benders algorithm in the context of two-stage integer programs. We do discuss, however, the generalization of the method to the case where integrality requirements are present in the two decision stages. Also, it is often the case that $B^s = B$, $\forall s \in S$. In fact, for the application we study later, this condition is true. However, for the present discussion, we stick to this more general form.

Given the structure of the optimization problem P, we note that with fixed values for the variables y, determining the optimal values for the variables x^s can be done by solving the following subproblem, $SP(y)_s$, for each scenario s:

$$z^s(y) = \text{minimize } c^\top x^s$$

subject to

$$Dx^s = d^s - B^s y,$$

$$x^s \geq 0.$$

With this definition of $z^s(y)$, we can reformulate P (Step 1 of the Benders decomposition approach) in the following way:

$$\text{minimize } f^\top y + \sum_{s \in S} p_s z^s(y)$$

subject to

$$Ay = b,$$

$$y \in \mathscr{Z}.$$

By taking the dual of $SP(y)_s$ we have, for each $s \in S$, the problem $DSP(y)_s$

$$\text{maximize } p^\top (d^s - B^s y)$$

subject to

$$p^\top D \leq c.$$

We note that the feasible region of DSP_s, $Q = \{p : p^\top D \leq c\}$, is the same for all scenarios. We assume that Q is feasible and has extreme points $q^i, i = 1, \ldots, I$, and extreme rays $w^j, j = 1, \ldots, J$. Given these extreme points and rays, a valid reformulation of P (Step 2 of the Benders decomposition approach), which we refer to as BP (i.e., the master problem), is

$$\text{minimize } f^\top y + \sum_{s \in S} p_s z^s \tag{5}$$

subject to

$$Ay = b, \tag{6}$$

$$q^{i\top}(d^s - B^s y) \leq z^s \quad \forall i = 1, \ldots, I, s \in S, \tag{7}$$

$$w^{j\top}(d^s - B^s y) \leq 0 \quad \forall j = 1, \ldots, J, s \in S, \tag{8}$$

$$y \in \mathcal{Z}. \tag{9}$$

Constraints (7) and (8) represent the optimality and feasibility cuts, respectively. Instead of solving problem (5)-(9) (Step 3 of the Benders decomposition approach), which would require enumerating all the extreme points and rays of Q, Benders decomposition repeatedly solves a relaxation, wherein only a subset of constraints (7) and (8) are considered (i.e., the master problem). Once a relaxation is solved to produce a vector $\bar{y}$, dual subproblems $DSP(\bar{y})_s$ are formed and solved to determine whether any optimality or feasibility cuts are violated. If so, they are added to the relaxation and the process repeats. Otherwise, the problem P has been solved. We summarize what is often referred to as the *Multi-cut* version of the L-shaped method in Algorithm 1, as developed in [4]. (Another version is the *Single-cut* version wherein in each iteration the scenario-based cuts are aggregated into a single cut, as was originally done in [37].)

If the second stage variables are integer, then $SP(y)_s$, $\forall s \in S$, represent integer programs to solve. In this case, standard duality cannot be applied to perform Step 2 of the decomposition approach. Instead, as developed in [5], general duality theory can be called upon to reformulate the subproblems using valid inequalities based on dual price functions to produce the optimality and feasibility cuts. It was shown in [5] how such functions can be derived when standard solution techniques are applied to solve the subproblems (i.e., cutting plane or branch and bound methods).

Algorithm 1 Benders decomposition

```
Create BP without constraints from sets (7) and (8)
while P not solved do
  Solve BP to get vector ȳ
  for s ∈ S do
    Solve dual subproblem DSP(ȳ)_s
    if constraints from sets (7) or (8) are violated then
      add them to BP
    end if
  end for
  if no violated constraints found then
    Stop {Solved P}
  end if
end while
```

In all cases, while Benders decomposition will converge to the optimal solution, the problem structure associated with the linking constraints (3) is lost. As a result, many of the valid inequalities that have been developed for the deterministic (single scenario) version are inapplicable. Furthermore, the relaxation of constraints (7)-(8) eliminates from BP all guiding information for y with respect to the second stage of the problem. Therefore, when the solution process begins, the solutions $\bar{y}$ obtained

may be arbitrarily poor with respect to their recourse cost. They may also be far from feasible in the second stage. Given that violated cuts are only introduced after the current relaxed *BP* is solved, the overall solution process can be excessively slow. In the next section, we present the different strategies that have been proposed to improve the Benders algorithm.

2.2 Improvements proposed for the algorithm

Geoffrion and Graves [18], were the first to develop an approach aimed at improving the Benders algorithm when applying it a multicommodity distribution problem. The authors observed that it is not necessary to solve the relaxed master problem to optimality to generate valid cuts. Actually, there is little incentive to do so at the beginning of the solution process (i.e., the relaxation is weak). Instead, the authors proposed to solve the problem at a given iteration of the algorithm to obtain a feasible solution that is within an optimality gap of ε. By fixing ε to appropriate values throughout the solution process (starting from higher values that are steadily decreased), cuts can be generated more effeciently. A similar idea can be applied to the scenario subproblems, as proposed by Zakeri, Philpott and Ryan [38]. The authors showed that suboptimal extreme points of the dual region of the scenario subproblems can be used to generate valid cuts. Again, by relaxing the requirement to solve systematically the subproblems to optimality, especially in the case of multistage stochastic programs, can considerably speedup the process by which cuts are generated by the L-Shaped algorithm.

Another strategy aimed at strengthening quickly the relaxed master formulation was proposed by McDaniel and Devine [25]. The authors showed that valid feasibility and optimality cuts can be obtained by solving the linear relaxation of the original problem. In effect, the Benders algorithm can be applied in a two phase approach to solve a mixed integer program. In the first phase, the linear relaxation of the original problem can be solved via the Benders algorithm. In doing so, a set of valid cuts are added to the relaxed master problem thus strengthening the model. In the second phase, the integrality constraints are reintroduced in the master's formulation and the solution process is applied anew. By proceeding in this way, fewer iterations in the second phase are usually necessary to converge to an optimal solution to the original problem.

In order to generate cuts faster, the master problem can also be solved heuristically. This idea was originally proposed by Côté and Laughton [9]. The authors suggested to apply Lagrangean relaxation to the optimality and feasibility cuts whenever the remaining constraint set in the master problem presents a special structure that is amenable to specialized algorithms. The problem can then be solved using any adapted heuristic and the solutions obtained can produce valid cuts. Although the original master problem still needs to be solved to optimality in order to ensure convergence of the algorithm, fewer of these solutions are usually needed. Related to this general idea, Rei *et al.* [29] and Poojari and Beasly [28] both proposed a Benders algorithm where multiple cuts are produced at each iteration by solving the master problem using respectively local

branching and a genetic procedure. Significant numerical improvements were reported in both cases when the algorithms were used to solve both deterministic [29, 28] and stochastic [29] integer programs.

Alternate formulations have also been proposed in replacement of the master problem. Cross decomposition, which was developed by Van Roy [31], falls into this category. This strategy is based on the simultaneous use of primal (or Benders) and dual (or Lagrangean) decompositions to solve mixed integer programs. Specifically, the author showed that solutions to the Lagrangean subproblem are feasible with respect to the Benders master problem and vice versa. Therefore, a sequence of solutions to the Benders master problem can be obtained by alternately solving the subproblems (i.e., Benders and Lagrangean). Using these solutions, valid cuts can be produced and added to strengthen the master's formulation. Again, convergence to optimality can only be maintained by solving on occasion the Benders master problem. However, by doing so less often, the solution process can be accelerated. Finally, Holmberg [19] studied the quality of the results obtained using different approximations for the Benders master problem. The author conducted a thorough analysis of the lower bounds attained by applying Lagrangean relaxation in the present context. The principal result of this study was to show that the bound defined by applying Lagrangean relaxation to the formulation of the Benders master is systematically worse than the one obtained by applying the same relaxation strategy to the original problem. This is the case even when all optimality and feasibility cuts are present in the master.

The general process by which cuts are added to the master problem has been improved as well. The studies conducted on this subject have either been based on defining strategies to select or strengthen the traditional cuts, or, on generating new valid cuts. Magnanti and Wong [23] were the first to propose an efficient strategy to select optimality cuts. The strategy developed finds non-dominated optimality cuts (i.e., Pareto-optimal cuts) whenever, for a given feasible solution to the master problem, there are multiple optimal solutions (i.e., extreme points) associated to the dual subproblem (or scenario subproblems in the stochastic case). In this context, among the dual optimal extreme points, the authors showed that a Pareto-optimal cut is one that produces a maximum value when evaluated at a core point of the feasible region of the master problem. As suggested in [23], in addition to the classical dual subproblem, a second one is solved to identify a non-dominated cut at a given iteration of the Benders algorithm. The objective function of this second subproblem is simply instantiated using a core point of the master and its feasible region is restricted to include only those extreme points which are optimal at the current iteration. The addition of non-dominated cuts can greatly improve the value of the lower bound obtained by the algorithm. However, to implement this approach, a master core point is needed and a second subproblem must be solved at each iteration.

As observed by Papadakos [27], the Magnanti and Wong method can be inefficient in cases where either the subproblem is hard to solve to optimality or core points to the master problem are difficult to obtain. Therefore, the author proposed two enhancements to the method. It was first shown that an independent subproblem (i.e., one that

is not restricted to the optimal extreme points) can be used to produce Pareto-optimal cuts. This result alleviates the necessity to solve two subproblems at each iteration of the Benders algorithm. The second enhancement was to propose alternative points that can be used as a proxy for the core ones. An efficient approach to produce these points was proposed for a special class of problems. An alternative cut selection strategy was developed by Fischetti, Salvagnin and Zanette [14]. Following this strategy, the subproblem is reformulated as a feasibility problem where cuts, both optimality and feasibility, are obtained by searching for minimal infeasible subsystems. Finally, all these strategies were further enhanced by Sherali and Lunday [36] who extended them to generate maximal non-dominated cuts. The authors showed that this can be achieved by simply perturbing the right-hand side values of the constraints of the subproblem.

As for strategies that have focused on strengthening the traditional cuts that are added by the Benders procedure, they apply the principle of iteratively generating multiple cuts (optimality or feasibility) that are tailored to include specific desirable characteristics. Saharidis and Ierapetritou [32] considered cases where optimality cuts are hard to obtain when applying the classical Benders approach. In such cases, the algorithm generates numerous feasibility cuts before obtaining a feasible solution to the problem that produces an optimality cut. To improve the Benders approach, the authors proposed to apply a maximum feasible subsystem (MFS) cut generation strategy. Following this strategy, each time a feasibility cut is added, an additional cut, referred to as the MFS cut, is also generated. This cut is obtained by solving an additional subproblem, which tries to determine the minimum number of constraints to relax in the original subproblem to obtain a feasible solution. Therefore, the MFS cut acts as an optimality one. A similar strategy was proposed by Saharidis, Minoux and Ierapetritou [33] who considered cases where the Benders algorithm tends to generate weaker low-density cuts (i.e., cuts that involve a small set of master problem variables). To solve this problem, a covering cut bundle (CCB) generation procedure was developed, to generate, at each iteration of the algorithm, a set of cuts (optimality or feasibility) covering all variables of the master problem. In a way, the CCB ensures a level of diversification in the cuts that are added to the master problem at each iteration.

New procedures to generate valid cuts have been proposed by Codato and Fischetti [6]who considered the special case where the Benders approach is applied to a binary program where the subproblem only involves testing for feasibility. In such a case, the traditional algorithm exclusively generates feasibility cuts. The authors showed that stronger valid cuts can be efficiently separated by searching for minimal infeasible subsystems in the solutions of the relaxed master problem. These combinatorial Benders cuts were shown to greatly improve the performance of the Benders algorithm on two classes of mixed-integer programs.

The majority of the techniques described so far have been developed in the context of applying Benders decomposition to solve deterministic integer programs. However, one also finds methods specifically tailored for two-stage stochastic integer programs. In cases where the second stage involves integer variables, Sen and Higle [34] proposed to apply disjunctive programming as a means to produce a convex characterization for

the subproblems. Specifically, the authors showed that valid inequalities, generated for a given solution to the master problem solution and a particular scenario subproblem, can be used to obtain valid inequalities for any other solution or subproblem. This result can then be applied to define the procedure by which cuts are generated in an overall Benders decomposition approach. This approach was further enhanced by Sen and Sherali [35] who illustrated how branch an cut algorithms can be efficiently used on the scenario subproblems.

As clearly illustrated in this section, methods aimed at improving the Benders algorithm have not been oriented towards modifying the decomposition strategy itself to render it more efficient. When the L-Shaped algorithm is applied to two stage stochastic programs, the projection involves always exclusively the first stage decisions. In doing so, the problem is decomposed for all scenarios used in the formulation. This has been the traditional approach applied given the general imperative of exploiting the special structure of the constraint set associated to these programs, while trying to keep a manageable master problem. However, by retaining some of the scenario subproblems in the master, one can directly alleviate the drawbacks associated with the L-Shaped algorithm. Furthermore, considering the steady advancements observed in the efficiency of off-the-shelf solvers, having a larger scaled master problem involved in the Benders solution process may remain computationally effective. This is the general idea behind our proposed partial decomposition strategy, which is the subject of the next section.

3 Partial decomposition

When trying to implement partial Benders decomposition for a two-stage stochastic integer program, there is an important methodological question to address: how should the scenario subproblems to retain be chosen? This section is dedicated to the partial decomposition strategies that we propose to provide answers to this general question. The section is divided into three subsections. In Section 3.1, we formalize the general concept of partial decomposition. The last two subsections are dedicated to the specific strategies that are developed. Section 3.2 describes the representation strategies, while Section 3.3 is devoted to the covering strategies. It should be noted that for the rest of this paper we assume that $B^s = B$, $\forall s \in S$, as is the case in the application we consider in our computational study. Thus, the defining characteristic of scenario s is the vector d^s.

3.1 General principles of partial decomposition

The general characteristic of our partial decomposition approach is that it is scenario-based. Specifically, assume we have identified a subset of scenarios $\bar{S} \subseteq S$ to be retained. When applying partial decomposition to the original problem (1)-(4), the vec-

tors $x^s \in \bar{S}$ are included in the set of complicating variables and the projection (i.e., Step 1 of the approach) is adapted accordingly. Therefore, once the dualization has occurred (i.e., Step 2 of the approach), we obtain problem $BP_{\bar{S}}$:

$$\text{minimize } f^\top y + \sum_{s \in \bar{S}} p_s c^\top x^s + \sum_{s \in S \setminus \bar{S}} p_s z^s \tag{10}$$

subject to

$$Ay = b, \tag{11}$$

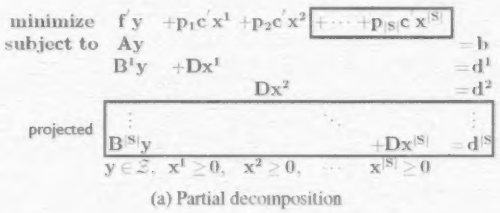
$$By + Dx^s = d^s, \quad \forall s \in \bar{S} \tag{12}$$

$$q^{i\top} d^s \leq q^{i\top} By + z^s \quad \forall i = 1, \ldots, I, s \in S \setminus \bar{S}, \tag{13}$$

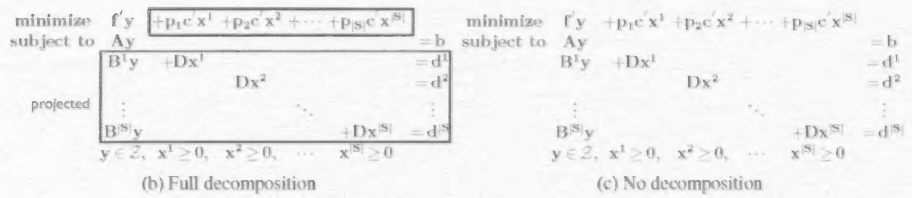
$$w^{j\top} d^s \leq w^{j\top} By \quad \forall j = 1, \ldots, J, s \in S \setminus \bar{S}, \tag{14}$$

$$y \in \mathcal{Z}, x^s \geq 0 \ \forall s \in \bar{S}. \tag{15}$$

We illustrate the concept of partial decomposition in Figure 1a and contrast it with a traditional, full decomposition (Figure 1b) and not decomposing a problem at all (Figure 1c).



(a) Partial decomposition


minimize f'y +p1c'x1 +p2c'x2 +··· +p|S|c'x|S|
subject to Ay = b
B1y +Dx1 = d1
Dx2 = d2
projected
B|S|y +Dx|S| = d|S|
y ∈ Z, x1 ≥ 0, x2 ≥ 0, ··· x|S| ≥ 0
minimize f'y +p1c'x1 +p2c'x2 +··· +p|S|c'x|S|
subject to Ay = b
B1y +Dx1 = d1
Dx2 = d2
B|S|y +Dx|S| = d|S|
y ∈ Z, x1 ≥ 0, x2 ≥ 0, ··· x|S| ≥ 0

(b) Full decomposition

(c) No decomposition

Figure 1: Decomposition approaches

An instance of $BP_{\bar{S}}$ can then be solved (i.e., Step 3 of the approach) via a slightly modified Benders solution procedure, which is represented by Algorithm 2. The only difference between the algorithms are in the initialization where, in Algorithm 2, the subset $\bar{S}$ and problem $BP_{\bar{S}}$ are created (i.e. the first two lines of Algorithm 2). Notice that the selection of scenarios included in $\bar{S}$ and the cardinality of the set influence the number of cuts generated.

There are several advantages in using $BP_{\bar{S}}$ when applying the Benders algorithm. First, the relaxation of the optimality and feasibility cuts is only applied to the constraints (13) and (14). Therefore, the inclusion of Constraints (12) in the master problem ensures that all its solutions are necessarily feasible with respect to the scenario subproblems associated to $s \in \bar{S}$. This can easily be verified by considering the result provided by Corollary 3.1, which is directly derived from duality theory. Partial decomposition therefore eliminates the need for the algorithm to generate feasibility cuts for the scenario subproblems $s \in \bar{S}$. Furthermore, by retaining the linking constraints (12) in the formulation of the master, some of the structure of the original problem is left intact. As a result, valid inequalities for the polyhedron $P_s = \{(x^s, y) : Ay = b, By + Dx^s = d, y \in \mathscr{Z}, x^s \geq 0\}$ can be added to the master for each $s \in \bar{S}$ to further strengthen the model.

Corollary 3.1. *If $(\bar{y}, \bar{x}^s, \forall s \in \bar{S}, \bar{z}^s, \forall s \in S \setminus \bar{S})$ is a feasible solution to model (10)-(12) and (15), then $(d^s - B\bar{y})^\top w^j \leq 0, \forall j = 1, \ldots, J, s \in \bar{S}$.*

Proof. The existence of the variable values $\bar{x}^s$, $\forall s \in \bar{S}$, implies that $SP_s(\bar{y})$ is feasible and thus, by weak duality, $DSP_s(\bar{y})$ can not be unbounded, $\forall s \in \bar{S}$. □

Partial decomposition can also help to improve the quality of the feasible solutions to the original problem (1)-(4) that are obtained throughout the Benders solution process. Given that $\bar{S} \subseteq S$, the law of total expectation entails that the expected recourse cost of a first stage solution y can be expressed as $\mathbb{E}(z^s(y) \mid s \in \bar{S}) \times P(\bar{S}) + \mathbb{E}(z^s(y) \mid s \in S \setminus \bar{S}) \times P(S \setminus \bar{S})$, where $P(\bar{S})$ and $P(S \setminus \bar{S})$ represent the probabilities of observing in the second stage a scenario in $\bar{S}$ and $S \setminus \bar{S}$, respectively. When partial decomposition is applied, the term $\mathbb{E}(z^s(y) \mid s \in \bar{S}) \times P(\bar{S})$ remains unchanged in the formulation of the master problem. Only the term $\mathbb{E}(z^s(y) \mid s \in S \setminus \bar{S}) \times P(S \setminus \bar{S})$ is dualized and then relaxed. Therefore, the approximation of the expected recourse cost provided by the objective function of the master problem is in this case more accurate than the one defined in the original decomposition approach. In turn, one can expect that Algorithm 2 will reach better solutions faster when compared to Algorithm 1. In addition, greater stability with respect to the solutions (and optimality cuts) obtained is also anticipated.

To instantiate partial decomposition, we now turn to the second issue identified previously, namely the construction of set $\bar{S}$. Randomly drawing scenarios to include in $\bar{S}$ is a straightforward approach that can be applied. However, in the next two subsections, we present systematic strategies that yield more effective partial decompositions. These strategies are categorized as either Representation or Covering. In each case, we present the general motivations behind the strategy and we describe how they are implemented. It should be noted that all strategies take as input the number of scenarios, K, to include in $\bar{S}$ and that they are based on the vectors d^s, each of which we assume has n elements.

Algorithm 2 Partial Benders decomposition

Create set $\bar{S} \subseteq S$
Create $BP_{\bar{S}}$ without constraints from sets (13) and (14)
while have not solved P **do**
  Solve $BP_{\bar{S}}$ to get vector $\bar{y}$
  for $s \in S \setminus \bar{S}$ **do**
    Solve dual subproblem $DSP(\bar{y})_s$
    if constraints from sets (13) or (14) are violated **then**
      add them to $BP_{\bar{S}}$
    end if
  end for
  if no violated constraints found **then**
    Stop {Solved P}
  end if
end while

3.2 Representation strategies

The motivation behind the proposed representation strategies is to choose a set $\bar{S}$ that best approximates a given characteristic of problem (1)-(4). What characteristic and approximation are considered gives rise to different approaches to construct set $\bar{S}$. In this subsection, we present two specific representation strategies. The first, which we refer to as the *Clustering-mean* strategy, considers the approximation provided by the scenarios in $\bar{S}$ of the probability distribution of the original set S. It is natural to expect that the closer one gets to approximating the original distribution by $\bar{S}$, the better the solutions obtained to the master problem will be throughout the solution process. The second strategy chooses instead a set of scenarios $\bar{S}$ such that the convex hull of vectors $d^s, s \in \bar{S}$, best approximates the convex hull of vectors $d^s, s \in S \setminus \bar{S}$. Proposition 3.2 clearly motivates this *Convex Hull* strategy.

Clustering-mean. This strategy begins by partitioning set S into K subsets of similar scenarios. *Similarity* between scenarios $s \in S$ and $s' \in S$ is measured as the Euclidean distance between their associated vectors d^s and $d^{s'}$. Then, to achieve a good approximation of the distribution of S by $\bar{S}$, the method chooses from each subset the scenario that is closest to its conditional mean as its representative to add to $\bar{S}$. We present this approach in Algorithm 3.

To obtain the partition of the scenario set S, a k-means clustering problem [24] is solved using the k-means++ [1] heuristic, which offers approximation guarantees. In all generality, the purpose of k-means clustering is to partition a set of data points into a given number of clusters such that each data point is included in the cluster whose mean is closest. To apply the k-means++ heuristic in the present context, each scenario $s \in S$ is represented by its associated vector d^s. The heuristic then partitions the $|S|$ scenarios into the subsets $S_1, \ldots, S_K$ (Line 2 of Algorithm 3). Finally, set $\bar{S}$ is defined

Algorithm 3 Choose $\bar{S}$ based on grouping similar scenarios

Require: Number of groups, K, to create
1: Set $\bar{S} = \emptyset$
2: Partition S into subsets $S_1, \ldots, S_K$ of similar scenarios with k-means++
3: **for all** sets S_i **do**
4: Calculate the mean, d^{S_i} of vectors $d^s, s \in S_i$
5: Add to $\bar{S}$ the scenario s in S^i such that d^s is closest to d^{S_i}
6: **end for**

as (Lines 3 to 6 of Algorithm 3):

$$\bar{S} = \bigcup_{i=1}^{K} \arg\min_{s \in S_i} \|d^s - d^{S_i}\|,$$

where function $\|\cdot\|$ is the Eucledean distance defined on the n dimensions of vectors d^s, $\forall s \in S$. For a more detailed presentation of this approach, we refer the reader to Crainic, Hewitt and Rei [10], who applied this method to a progressive hedging-based meta-heuristic.

Convex Hull. An alternate way of viewing representation is through the pursuit of convex hulls in integer programming, as having a representation of the convex hull of the feasible region reduces the complexity of solving the resulting integer program. Following this approach, the scenarios added to $\bar{S}$ are the ones that include in their associated convex hull the scenarios in $S \setminus \bar{S}$. The following proposition shows the value of this idea in the present context:

Proposition 3.2. *Consider $\bar{S} \subseteq S$ and $s' \in S \setminus \bar{S}$, such that $\exists\, \alpha_s^{s'} \geq 0, s \in \bar{S}$: $\sum_{s \in \bar{S}} \alpha_s^{s'} = 1$ and $\sum_{s \in \bar{S}} d^s \alpha_s^{s'} = d^{s'}$. If $(\bar{y}, \bar{x}^s, \forall s \in \bar{S}, \bar{z}^s, \forall s \in S \setminus \bar{S})$ is a feasible solution to model (10)-(12) and (15), then $w^{j\top} d^{s'} \leq w^{j\top} B\bar{y}, \forall j = 1, \ldots, J$.*

Proof. We have by Corrolary 3.1 that the constraints $w^{j\top}(d^s - B\bar{y}) \leq 0$ are satisfied $\forall j = 1, \ldots, J, s \in \bar{S}$. Given $s \in \bar{S}$ and $s' \in S \setminus \bar{S}$, we have that

$$\begin{aligned} w^{j\top} d^{s'} &= w^{j\top} \Big(\sum_{s \in \bar{S}} d^s \alpha_s^{s'}\Big) \\ &= \sum_{s \in \bar{S}} \alpha_s^{s'} (w^{j\top} d^s) \\ &\leq \sum_{s \in \bar{S}} \alpha_s^{s'} (w^{j\top} B\bar{y}) \\ &= w^{j\top} B\bar{y}. \end{aligned}$$

We can thus conclude that $w^{j\top} d^{s'} \leq w^{j\top} B\bar{y}, \forall j = 1, \ldots, J$. □

Thus, if there exists a convex combination defined on the vectors d^s, for $s \in \bar{S}$, that can express the vector $d^{s'}$, for a given $s' \in S \setminus \bar{S}$, then a feasible solution to the partially decomposed master problem (i.e., $\bar{y}$) necessarily induces a feasible scenario subproblem for scenario s' (i.e., $DSP(\bar{y})_{s'}$ is bounded). In this case, there will be no feasibility cuts (14) generated for $SP(y)_{s'}$ when executing Algorithm 2. Therefore, the more scenarios in $S \setminus \bar{S}$ can be expressed as convex combinations of the scenarios in $\bar{S}$, the less feasibility cuts need to be generated for the algorithm to converge. Although it may be difficult to obtain perfectly defined combinations, Proposition 3.2 provides criteria to efficiently choose the scenarios to include in $\bar{S}$.

To construct set $\bar{S}$, we propose to solve a mixed integer program. This program includes the following decision variables: q_s, $\forall s \in S$, are binary variables that indicate whether scenario s is included in $\bar{S}$; $\alpha_s^{s'}$, $\forall s, s' \in S$, are continuous variables that model how the vector $d^{s'}$ can be approximated with a convex combination of the scenarios $s \in S$ such that $q_s = 1$; r_l^s, $\forall s \in S, l = 1, \ldots, n$, are continuous variables that represent how element l of vector d^s is represented by a convex combination of the scenarios $s \in S$ such that $q_s = 1$; and e_l^s, $\forall s \in S, l = 1, \ldots, n$, are continuous variables that measure the error in that representation. Specifically, we solve the mixed integer program

$$\text{minimize} \sum_{s \in S} \sum_{l=1}^{n} e_l^s$$

subject to

$$r_l^{s'} = \sum_{s \in S} \alpha_s^{s'} d_l^s, \ \forall s' \in S, l = 1, \ldots, n, \tag{16}$$

$$\alpha_s^{s'} \leq q_s, \ \forall s \in S, s' \in S, \tag{17}$$

$$\sum_{s \in S} \alpha_s^{s'} = 1, \ \forall s' \in S, \tag{18}$$

$$\sum_{s \in S} q_s \leq K, \tag{19}$$

$$e_l^s \geq r_l^s - d_l^s, \ \forall s \in S, \ l = 1, \ldots, n, \tag{20}$$

$$e_l^s \geq d_l^s - r_l^s, \ \forall s \in S, \ l = 1, \ldots, n, \tag{21}$$

$$q_s \in \{0, 1\}, \ \forall s \in S, \ \alpha_s^{s'} \geq 0, \ \forall s \in S, s' \in S. \tag{22}$$

The objective is to minimize the total error associated with the representation r^s and the vector d^s associated with each scenario. The dimensional representation values $r_l^{s'}$ are formulated in (16), while Constraints (17) enforce that the weights $\alpha_s^{s'}$ only take on positive values when scenario s is chosen to be included in $\bar{S}$. Constraints (18), coupled with the second variable definition in constraints (22), ensure that for each $s' \in S$ the weights $\alpha_s^{s'}$ constitute a convex combination. Constraint (19) limits the number of scenarios that are included in $\bar{S}$ to be at most K. Constraints (20) and (21), coupled with the objective, define that $e_l^s = |r_l^s - d_l^s|$. Constraints (22) impose the necessary non-negativity and integrality requirements on the decision variables.

3.3 Covering strategies

The principle behind the covering strategies proposed is to retain scenario subproblems that ensure feasibility for projected ones; as we will see later on, this can be ensured for scenarios s, s' when $d^s \geq d^{s'}$. Let us first define the following concepts:

Definition 3.3. *Given two scenarios $s \in \bar{S}$ and $s' \in S \setminus \bar{S}$, if an element l that is such that $d_l^s \geq d_l^{s'}$, we state that s covers s' with respect to l. If s covers s' with respect to all elements $l = 1, \ldots, n$, then we declare that scenario s covers s'.*

The motivating result behind the proposed covering strategies is then contained in the following proposition:

Proposition 3.4. *Consider $s \in \bar{S}$ and $s' \in S \setminus \bar{S}$ such that s covers s'. If $(\bar{y}, \bar{x}^s, \forall s \in \bar{S}, \bar{z}^s, \forall s \in S \setminus \bar{S})$ is a feasible solution to model (10)-(12) and (15), then $w^{j\top} d^{s'} \leq w^{j\top} B\bar{y}, \forall j = 1, \ldots, J$ such that $w^j \geq 0$.*

Proof. Given $s \in \bar{S}$, $s' \in S \setminus \bar{S}$ and $w^j \geq 0$, we have that $w^{j\top} d^{s'} \leq w^{j\top} d^s$, considering that $d_l^s \geq d_l^{s'}, l = 1, \ldots, n$ by the assumption that s covers s' (i.e., Definition 3.3). By Corrolary 3.1, which etails that the constraints $w^{j\top}(d^s - B\bar{y}) \leq 0$ are satisfied $\forall j = 1, \ldots, J, s \in \bar{S}$, we thus obtain that $w^{j\top} d_l^{s'} \leq w^{j\top} d_l^s \leq w^{j\top} B\bar{y}$, $\forall j = 1, \ldots, J$ such that $w^j \geq 0$. □

Proposition 3.4 is again based on the principle of enabling feasibility in specific non-retained scenario subproblems. If a scenario $s' \in S \setminus \bar{S}$ is covered by a scenario $s \in \bar{S}$, then a first stage solution to the partially decomposed master problem (i.e., $\bar{y}$) is such that a subset of feasbility cuts associated with $SP(\bar{y})_{s'}$ are necessarily enforced (i.e., $w^{j\top} d^{s'} \leq w^{j\top} B\bar{y}, \forall j = 1, \ldots, J$ such that $w^j \geq 0$). Therefore, the more scenarios in $S \setminus \bar{S}$ are covered by scenarios in $\bar{S}$, the less feasibility cuts need to be generated by Algorithm 2 to converge. We next describe two strategies for choosing the scenarios to include in $\bar{S}$ that are based on this idea of covering. We refer to these strategies as *Clustering-greatest* and *Row Covering*.

Clustering-greatest. As in the case of the *Clustering-mean* approach, the present strategy first partitions the set S into subsets of similar scenarios using the k-means++ heuristic (i.e., Step 2 of Algorithm 3). We again represent by $S_i, i = 1, \ldots, K$ the clusters of scenarios that define the partition of S. As in the previous approach, representative scenarios for the subsets obtained are then chosen to be included in set $\bar{S}$. However, in this case, the scenario that is chosen as the representative is the one that covers the maximum number of elements throughout the cluster. Therefore, considering a scenario $s \in S$, an element $l = 1, \ldots, n$, and a cluster $S_i \subseteq S$, let us define the following function:

$$\delta_l^s(S_i) = \begin{cases} 1 & \text{If } d_l^s \geq d_l^{s'}, \forall s' \in S_i, s \neq s' \\ 0 & \text{Otherwise.} \end{cases}$$

Function $\delta_l^s(S_i)$ takes value 1 if scenario s covers all scenarios contained in cluster S_i with respect to element l. We can now define the total number of elements covered by s in S_i in the following way:

$$G^s(S_i) = \sum_{l=1}^{n} \delta_l^s(S_i).$$

Using the partition of S that is provided by the clusters S_i, $i = 1,\ldots,K$, we construct subset $\bar{S}$ as

$$\bar{S} = \bigcup_{i=1,\ldots,K} \arg\max_{s \in S_i} G^s(S_i).$$

Row Covering. The previous strategy, when choosing $\bar{S}$, ignores whether a single element $d_l^{s'}$, for $s' \in S \setminus \bar{S}$, is covered by multiple scenarios in $\bar{S}$. Thus, the *Row Covering* strategy focuses instead on the total number of distinct elements d_l^s that are covered. To do so, we again propose to solve an integer program to construct $\bar{S}$. Let us first define the values $\delta_l^{ss'}$, $\forall s, s' \in S$ such that $s \neq s'$, and $l = 1,\ldots,n$, as follows:

$$\delta_l^{ss'} = \begin{cases} 1 & \text{If } d_l^s \geq d_l^{s'} \\ 0 & \text{Otherwise.} \end{cases}$$

Therefore, value $\delta_l^{ss'}$ is 1 if scenario s covers scenario s' with respect to element l. To formulate the integer program proposed, we now define the binary variables q_s, $\forall s \in S$, and $b_l^{s'}$, $\forall s' \in S$ and $l = 1,\ldots,n$. Variable q_s expresses whether the scenario s is added to $\bar{S}$ or not, while variable $b_l^{s'}$ indicates whether or not the element $d_l^{s'}$ is covered by one of the scenarios in $\bar{S}$. The following mixed integer program is then solved to obtain $\bar{S}$:

$$\text{maximize} \sum_{s' \in S} \sum_{l=1}^{n} b_l^{s'}$$

subject to

$$b_l^{s'} \leq \sum_{s \in S} \delta_l^{ss'} q_s, \ \forall s' \in S, l = 1,\ldots,n, \tag{23}$$

$$\sum_{s \in S} q_s \leq K, \tag{24}$$

$$q_s \in \{0,1\}, \forall s \in S, b_l^{s'} \in \{0,1\}, \forall s' \in S, l = 1,\ldots,n. \tag{25}$$

The objective of this optimization problem is to maximize the number of elements d_l^s that are covered. Constraints (23) and the objective ensure that the variables $b_l^{s'}$ take on the value 1 if a scenario s is included in $\bar{S}$ such that $d_l^s \geq d_l^{s'}$. The constraint (24) limits the number of scenarios that are included in $\bar{S}$ to be at most K. Finally, constraints (25) impose the integrality requirements on the decision variables.

4 Numerical results

To resolve the issues of whether a partial decomposition should be performed and, if so, how it should be performed, we complement the analysis presented in the previous section with results from an extensive computational study. We begin in Section 4.1 with a detailed discussion of the setting for our experiments. Then, in Section 4.2, we resolve in the affirmative the issue of whether a partial decomposition should be performed. We study next, Section 4.3, which of the strategies presented in Sections 3.2 and 3.3 yields the most effective partial decomposition. Finally, we analyze in Section 4.4 how partial decomposition impacts various metrics related to the performance of Benders method.

4.1 Experimental setting

In this section, we describe the specific problem used to perform our numerical analysis, the characteristics of the test instances, and how the algorithms tested were implemented.

The Problem. We selected the stochastic fixed charge multi-commodity network design problem to study the effectiveness of the partial decomposition strategies proposed for the Benders algorithm. Two reasons for this choice: these problems naturally appear in many applications (e.g., [22, 21]), and they are notoriously hard to solve (e.g., [12, 11]). Consider a directed network with node set N, arc set A, commodity set K, and scenario set S. The formulation of the stochastic fixed charge multi-commodity network design problem, *CMND(S)*, is

$$\text{minimize} \sum_{(i,j)\in A} f_{ij}y_{ij} + \sum_{s\in S} p_s\Big(\sum_{k\in K}\sum_{(i,j)\in A} c_{ij}^k x_{ij}^{ks}\Big) \tag{26}$$

subject to

$$\sum_{j\in N^+(i)} x_{ij}^{ks} - \sum_{j\in N^-(i)} x_{ji}^{ks} = d_i^{ks} \quad \forall i\in N, \forall k\in K, \forall s\in S, \tag{27}$$

$$\sum_{k\in K} x_{ij}^{ks} \le u_{ij}y_{ij} \quad \forall (i,j)\in A, \forall s\in S, \tag{28}$$

$$y_{ij}\in\{0,1\} \quad \forall (i,j)\in A, \tag{29}$$

$$x_{ij}^{ks}\ge 0 \quad \forall (i,j)\in A, \forall k\in K, \forall s\in S, \tag{30}$$

where, y_{ij} indicates whether arc $(i,j)\in A$ is selected (i.e., installed in the network) in the first stage of the problem, and f_{ij} is the cost (often called the fixed charge) of including arc (i,j) in the network. In the second stage of the problem, the obtained network is used to flow the commodities to meet the observed demands. Variable x_{ij}^{ks} is the amount of the demand of commodity $k\in K$ that flows on arc (i,j), considering that scenario $s\in S$ is observed in the second stage of the problem, c_{ij}^k being the cost per unit of demand k flowed on arc (i,j). Constraints (27) are flow-conservation equations

ensuring that each commodity's demand may be routed from its origin node to its destination node in each scenario s. Therefore, assuming that v^{ks} is the volume of commodity k in scenario s, the parameter d_i^{ks} is either set to v^{ks} if node i is the origin of the commodity k, $-v^{ks}$ if node i is the destination of the commodity k, or 0 otherwise. Constraints (28) guarantee that the same design is used in each scenario, and that the arc capacity (u_{ij}) is never violated. Finally, Constraints (29) and (30) impose the necessary integrality and non-negativity requirements on the decision variables of the model.

While Proposition 3.4 does not apply to the $CMND(S)$ (the constraints associated with the vector d^s are equality constraints and hence, the dual variables are not restricted to be non-negative), the conclusion still holds. Namely, when $s \in \bar{S}$ and $s' \in S \setminus \bar{S}$ are such that s covers s' and $(\bar{y}, \bar{x}^s, \forall s \in \bar{S}, \bar{z}^s, \forall s \in S \setminus \bar{S})$ is a feasible solution to $CMND(\bar{S})$ then $w^{j\top} d^{s'} \leq w^{j\top} B\bar{y}$. Or, if y induces a design that enables the routing of commodity demands for scenario s then it will also enable the routing of commodity demands for scenario s'.

The Instances. We use 7 instance classes (4-10), for our computational study, taken from the set of R instances seen in [12]. Attributes of each class are given in Table 1. Each of these classes contains five networks, labeled 1,3,5,7, 9, yielding a total of 35 networks. The labels 1,3,5,7, and 9 reflect an increasing ratio of fixed to variable costs and total demand to capacity. A detailed description of the instances can be found in [11].

Class	$\|N\|$	$\|A\|$	$\|K\|$
4	10	60	10
5	10	60	25
6	10	60	50
7	10	82	10
8	10	83	25
9	10	83	50
10	20	120	40

Table 1: Instance Class Characteristics

Characteristic	Values
$\|S\|$	16,32,64
% of d_i^s that are positively correlated	0, 20, 40, 60, 80

Table 2: Scenario Characteristics

Then, we generate several instances with varying number of scenarios for each network in each instance class. We detail data regarding the scenarios in Table 2. The first line indicates that, for each of the 35 networks, there are five instances with 16 scenarios, five with 32 scenarios, and five with 64 scenarios. The five instances for each network vary with respect to correlation. Specifically, as shown on the second line, for each network and number of scenarios, each instance of the five is characterized by a different fraction of the elements $d_i^s, i = 1, \ldots, n$ that are positively correlated. In summary, we have 525 instances of varying network structure, number of scenarios, and correlation.

The Implementations. In all experiments, we executed our implementations of Benders decomposition for at most two hours on a cluster of machines with 8 Intel Xeon CPUs running at 2.66 GHz with 32 GB RAM. All linear and mixed integer programs

were solved with CPLEX 12. We solved $CMND(S)$instances with either CPLEX or a Benders-based algorithm, which uses either a partial or a full decomposition. All algorithms were executed with an optimality gap tolerance of 1%. All computation times reported are in seconds.

We have implemented a Benders algorithm that includes many of the enhancements developed for the original solution procedure. We implemented the *Multi-cut* version of the L-Shaped method [4], as previously presented in Algorithm 1 in Section 2.1. Preliminary tests showed that this version of the method outperformed the original *Single-cut* version [37] on the instances used. It has often been observed that the lack of structure in the master problem hampers the ability of a Benders implementation to solve instances in reasonable run-times. Therefore, we also opted for the two phase Benders solution approach developed in [25]. As such, the first phase involves the solution of the linear relaxation of problem (26)-(30) via Algorithm 1 or Algorithm 2 when partial decomposition is used. The cuts collected while solving the linear relaxation are used to strengthen the formulation of the master problem solved in the second phase, wherein the integrality constraints (29) are reintroduced and the Benders algorithm is again applied to produce an optimal solution to the $CMND(S)$.

Regarding the second phase, the master problem is solved via CPLEX, with optimality and feasibility cuts added at nodes of the branch-and-bound tree whenever integer feasible solutions are found. This approach is inspired by the strategy proposed in [18] where suboptimal solutions are used to generate cuts. It is also similar to the hybrid method proposed by Hooker and Ottosson [20], which combines Benders decomposition and constraint programming to solve a larger class of problems. Finally, as local branching was shown to speed up the execution of Benders decomposition [29], we turn on CPLEX's implementation of local branching when solving a master problem in the second phase of the algorithm,

Regarding the generation of optimality and feasibility cuts, we have implemented the approach originally proposed in [23], guaranteeing that only non-dominated cuts are added to the master problem. Also, problem-specific inequalities for the CMND(S) are added to further strengthen the formulation of the master problem. Specifically, we add inequalities of the form $\sum_{j \in N^+(i)} y_{ij} \geq 1$, where node i is the origin for some commodity k. Similar inequalities can be added for destination nodes for commodities. Both types of inequalities are included in the master's formulation at the beginning of the first phase of the algorithm. Furthermore, it is well known that, when $d^{ks} < u_{ij}$, adding constraints of the form $x_{ij}^{ks} \leq d^{ks} y_{ij}$ to (26)-(30) greatly strengthens the formulation. At the same time, even in a deterministic setting (where $|S| = 1$), there are often too many of these inequalities to add them beforehand. Consequently, it is necessary to add them dynamically in a cutting plane algorithm fashion [26]. When partial decomposition is applied, by retaining some of the variables x_{ij}^{ks}, we therefore dynamically add these inequalities when solving the linear relaxation of the $CMND(S)$. The collected inequalities $x_{ij}^{ks} \leq d^{ks} y_{ij}$ are then kept in the master's formulation when the second phase of the algorithm is performed.

4.2 Benchmarking the use of partial decomposition

We first study whether a partial decomposition should be used. To do so, we compare the performances of a partial decomposition with the scenarios in $\bar{S}$ drawn randomly from S (labeled *Random* in the following tables), of a full decomposition (labeled *Traditional*), and of not performing a decomposition at all (labeled *CPLEX*) wherein we simply solve the instance of CMND(S) as a MIP. The results for the Random strategy are based on executing the algorithm five times (each with a different seed for the random number generator) for each instance. We consider two sizes for the set $\bar{S}$, 4, and 8.

We report in Tables 3a and 3b, by method, the average of the total time the algorithm executed (*Total Time*) and the final gap it produced (*Gap*), measured as 100*(Primal-Dual)/Primal, where Primal represents the value of the primal solution found by the algorithm and Dual is the value of the dual bound produced. We also report the percentage of instances the method was able to solve to within 1% (*Solved*). We report results for all instances with 64 scenarios in Table 3a and for instances with 64 scenarios that CPLEX could not solve in Table 3b. We see that, with respect to the optimality gap produced, a decomposition-based approach is superior to CPLEX for these instances and that performing a partial decomposition is better than performing a full decomposition. We will return to the question of why fewer instances are solved when performing a partial decomposition in Section 4.4. As can be seen in Table 3b, CPLEX performs poorly on the instances it could not solve, and performing a partial decomposition leads to the smallest optimality gap on those same instances.

Strategy	$\lvert\bar{S}\rvert$	Total Time	Gap	Solved
CPLEX	64	3,067.47	11.48%	60.00%
Traditional	0	3,129.62	8.31%	52.57%
Random	4	3,915.27	6.57%	45.71%
	8	3,993.82	6.13%	45.14%

(a) All 175 instances

Strategy	$\lvert\bar{S}\rvert$	Total Time	Gap	Solved
CPLEX	64	7,200.00	35.17%	0.00%
Traditional	0	4,317.73	16.88%	23.64%
Random	4	6,030.59	13.04%	13.09%
	8	6,118.93	12.08%	13.45%

(b) 70 instances CPLEX could not solve

Table 3: Instances with $|S| = 64$

4.3 Benchmarking partial decomposition strategies

Having established that a partial decomposition is worth performing, we next benchmark the strategies for producing a partial decomposition discussed in Sections 3.2 and 3.3. These strategies require a parameter, K, which in turn dictates the size of the set $\bar{S}$, or, the number of scenarios we retain. We consider three sizes for the set $\bar{S}$, 1, 4, and 8. For *Clustering-mean*, choosing one scenario to retain (i.e., $|\bar{S}| = 1$) results in choosing the scenario that is closest to the mean of all scenarios in S. Because retaining only one scenario for the *Convex Hull* strategy is not meaningful, we do not do it.

We report in Table 4 results for instances with 16, 32, and 64 scenarios. We see that, given a size for $\bar{S}$, every strategy for constructing a set $\bar{S}$ of that size results in a smaller gap (and almost always less time) than the full decomposition scheme employed by the Traditional implementation. While the Random strategy is effective, nearly all the other strategies yield a lower gap and often terminate in less time.

Recalling our categorization of strategies, we note that the Covering strategies perform better than the Representation ones, yielding lower gaps and total times, and a higher percentage of solved instances. We conclude that Convex Hull is the best Representation strategy and that the two Covering strategies are nearly equivalent. We note that, when $|\bar{S}| = 1$, the Row Covering strategy lowers the gap by more than 2% from the Traditional one for all values of $|S|$. Given the relative performance of the various strategies, we next restrict our analysis to the strategies Traditional, Random, Convex Hull, and Row Covering.

		$\|S\| = 16$			$\|S\| = 32$			$\|S\| = 64$		
Strategy	$\|\bar{S}\|$	Total Time	Gap	Solved	Total Time	Gap	Solved	Total Time	Gap	Solved
Traditional	0	2,941.84	6.90%	54.29%	2,972.78	7.68%	53.71%	3,129.62	8.31%	52.57%
Random	1	3,331.64	5.50%	57.71%	3,766.17	5.29%	51.60%	3,950.03	6.30%	45.71%
	4	3,519.08	4.53%	56.11%	3,779.44	5.22%	50.74%	3,915.27	6.57%	45.71%
	8	3,700.15	4.24%	52.00%	3,787.80	5.21%	50.97%	3,993.82	6.13%	45.14%
Clustering-mean	1	3,504.08	4.00%	57.14%	3,785.85	5.07%	50.86%	4,008.87	6.14%	45.14%
	4	3,146.16	3.36%	65.71%	4,257.03	4.85%	49.71%	4,501.13	6.53%	42.29%
	8	2,351.89	2.11%	73.14%	3,861.60	4.86%	52.57%	4,587.91	6.60%	40.57%
Convex Hull	4	2,983.58	2.92%	66.29%	3,713.28	4.37%	52.57%	4,088.00	5.84%	41.14%
	8	2,117.62	1.94%	78.29%	3,508.38	3.70%	56.00%	4,090.12	5.59%	42.29%
Clustering-greatest	1	3,590.98	3.98%	60.57%	3,997.52	4.94%	50.86%	4,350.67	6.22%	44.00%
	4	2,857.73	2.59%	67.43%	3,806.83	3.66%	58.29%	4,435.51	4.88%	43.43%
	8	1,898.25	1.68%	80.57%	3,589.48	2.93%	62.29%	4,465.22	4.75%	43.43%
Row Covering	1	3,551.08	4.02%	61.43%	3,956.86	4.90%	51.43%	4,347.79	6.21%	44.29%
	4	2,759.10	2.60%	69.71%	3,695.79	3.54%	57.14%	4,233.17	4.89%	41.71%
	8	1,848.51	1.67%	81.14%	3,357.26	3.03%	62.86%	4,175.83	4.61%	45.14%

Table 4: Performance by strategy and $|\bar{S}|$.

While Table 4 reports averages across all correlation levels, we next present in Figure 2 the average optimality gap reported by strategy for each correlation level. For the partial decomposition strategies, we average over results when $|\bar{S}| = 4$. We see that for all correlation levels, performing a partial decomposition leads to a much smaller optimality gap than not doing so, and that performing a partial decomposition based on the Row Covering strategy always yields the smallest optimality gap. We also note that the optimality gap produced is much less sensitive to the correlation level when performing a partial decomposition compared to when one is not performed.

The Row Covering strategy involves first solving an optimization problem with an objective function that serves as a proxy for the performance of an algorithm such as Algorithm 2. We study the effectiveness of this proxy in Table 5, the values reported being averages across results for all instances (i.e., all values of $|S|$ and all correlation levels). Specifically, we report in row *% Elements covered* an average of the quantity $|\{(i,s) \;\; i = 1,\ldots,n; s \in S : v_i^{\bar{s}s} \text{ for some } \bar{s} \in \bar{S}\}|/(|S| * n)$, or the percentage of (element, scenario) pairs that are covered by a scenario that was retained in set $\bar{S}$. We report in row *% of instances* the percentage of instances for which the preceding statistic equals the corresponding value in the % Elements covered row. Next, we report in rows *Gap* and *Feasibility cuts* averages of these statistics for experiments with the same percentage

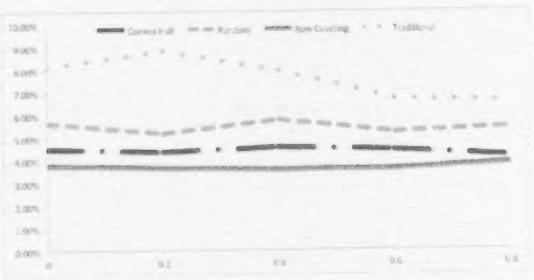


Figure 2: Optimality gap by correlation level

covered. We see that, more often than not, the better the set $\bar{S}$ does at covering the remaining scenarios, the better the performance of the partial decomposition, with a sharp breakpoint occurring at 95%. We also note that we are often able to cover 95% or more elements.

% Element covered	89	90	91	92	93	94	95	96	97	98
% of instances	2.86%	8.67%	11.43%	16.19%	17.14%	9.52%	4.76%	15.24%	11.43%	4.76%
Gap	5.38%	7.27%	5.90%	5.51%	3.75%	3.66%	1.02%	1.82%	0.84%	0.84%
Feasibility cuts	54.13	44.54	40.90	41.52	52.83	30.32	18.44	20.19	18.82	15.41

Table 5: Performance of *Row Covering* by % covered.

Finally, in Tables 6a and 6b, we repeat Tables 3a and 3b adding entries for the Row Covering strategy. We see that this strategy leads to a significantly smaller gap than the other methods. We will discuss the fact that performing a full decomposition leads to a greater number of solved instances next as we study various performance statistics related to performing partial decomposition.

Strategy	$\lvert\bar{S}\rvert$	Total Time	Gap	Solved
CPLEX	64	3,067.47	11.48%	60.00%
Traditional	0	3,129.62	8.31%	52.57%
Random	4	3,915.27	6.57%	45.71%
	8	3,993.82	6.13%	45.14%
Row Covering	4	4,233.17	4.89%	41.71%
	8	4,175.83	4.61%	45.14%

(a) All 175 instances

Strategy	$\lvert\bar{S}\rvert$	Total Time	Gap	Solved
CPLEX	64	7,200.00	35.17%	0.00%
Traditional	0	4,317.73	16.88%	23.64%
Random	4	6,030.59	13.04%	13.09%
	8	6,118.93	12.08%	13.45%
Row Covering	4	6,563.13	9.78%	5.45%
	8	6,509.80	9.93%	3.64%

(b) 70 instances CPLEX could not solve

Table 6: Instances with $|S| = 64$

4.4 Analyzing the impact of partial decomposition

We next seek a fuller picture of the performance seen when performing partial decomposition. All the results presented in this subsection are based on averaging across

results for all instances (i.e., all values of $|S|$ and all correlation levels). For partial decomposition strategies, results achieved with $|\bar{S}| = 4$ are reported.

Recall that, in our Benders implementation, we first solve the linear programming relaxation of P with Algorithm 1. After doing so, we solve P, augmented with the cuts found while solving its linear relaxation, with a branch-and-bound-based algorithm wherein cuts are generated throughout the tree search. Notice that we define an iteration of Algorithm 1 as one execution of the while loop, wherein a relaxation of the problem is solved and violated cuts are generated.

Then, to measure the impact of partial decomposition on convergence, we report in Table 7 a comparison of the number of iterations required (on average) to solve the linear programming relaxation of P for each strategy. While it is to be expected that performing a partial decomposition speeds up convergence (cuts never need to be generated for scenarios that are retained), the magnitude of the speed-up, even over performing a Random partial decomposition, is surprising.

	Traditional	Random	Convex Hull	Row Covering
Iterations to converge	186.93	56.90	18.19	16.22

Table 7: Convergence rate by strategy

One statistic related to convergence is the number of Benders cuts generated. We next report in Table 8, by correlation level, the impact of the different strategies on the number of Optimality (*Opt. cuts*) and Feasibility (*Feas. cuts*) cuts generated. We report the average number of cuts generated for each strategy, considering both the cuts generated when solving the linear programming relaxation of P with Algorithm 1 and when solving P via a branch-and-bound-based Benders implementation.

Correlation	Traditional		Random		Convex Hull		Row Covering	
	Opt. cuts	Feas. cuts	Opt. cuts	Feas. cuts	Opt. cuts	Feas. cuts	Opt. cuts	Feas. cuts
0	9,099.02	4,990.44	5,831.07	798.82	3,531.32	141.02	3,324.13	41.26
0.2	9,518.81	4,543.37	6,509.87	833.62	3,745.46	186.91	3,332.35	39.92
0.4	9,473.58	4,550.91	6,463.41	1,012.70	3,734.56	197.26	3,273.38	30.10
0.6	9,732.41	4,128.62	6,517.09	857.83	3,724.43	129.02	3,265.50	30.30
0.8	9,567.20	3,951.27	6,397.28	878.79	3,608.57	142.42	3,200.87	31.90
Average	9,478.20	4,432.92	6,343.74	876.35	3,668.87	159.33	3,279.25	34.70

Table 8: Optimality and feasibility cuts by strategy and correlation

We remark that performing a partial decomposition yields a significant decrease in the number of feasibility and optimality cuts generated, and that the Convex Hull and Row Covering strategies perform much better than choosing scenarios to retain randomly. The Row Covering strategy outperforms the Convex Hull strategy, reducing the number of optimality cuts by over 10% and the number of feasibility cuts to less than fifty for each correlation level. The low number of feasibility cuts generated when performing the Row Covering strategy echoes Proposition 3.4. And, not surprisingly, the number of feasibility cuts found when performing the Row Covering strategy trends down as the correlation level increases.

One computational issue that is often encountered when employing Benders is instability; namely that the values of the first stage variables y fluctuate wildly from one iteration to the next. We next study whether performing a partial decomposition mitigates this issue by measuring the Hamming distance between successive vectors of first stage variable values (Δ *next*), which represent a network design in our application. We report averages of these distances in Table 9 along with the Hamming distance from each design to the final design produced (Δ *final*). We see that the Row Covering strategy outperforms both Traditional and Random with respect to stability, as the designs change the least in the course of solving P and are the closest to the final design produced.

	Traditional	Random	Convex Hull	Row Covering
Δ next	23.43	21.30	16.97	16.21
Δ final	16.76	14.16	12.71	11.85

Table 9: Stability (measured by Hamming distance) by strategy

We next study the performance of each strategy with respect to the quality of the primal solution and dual bound produced while solving the LP relaxation of P and at termination. By attempting to solve each instance with CPLEX (executed with a time limit of two hours), we have a fixed dual bound that we call *CPLEX LB* against which we measure. With *Primal* representing the value of the best primal solution produced by a strategy for an instance, we calculate the *Primal gap* for that solution as 100*(Primal-CPLEX LB)/Primal. Similarly, while solving the linear programming relaxation of P with a decomposition-based approach, a feasible solution *LP Primal* can be produced through a rounding procedure. We calculate the gap for that solution *(LP Primal gap)* as 100*(LP Primal - CPLEX LB)/LP Primal. We report averages of these gaps in Table 10. Similarly, we compare for each strategy the dual bound produced when solving the LP relaxation and at termination against CPLEX LB. Thus, with *LP Dual* representing the bound produced when the LP relaxation of P is solved, we calculate *LP dual gap* as 100*(CPLEX LB-LP Dual)/CPLEX) LB. Finally, with *Dual* representing the bound produced at termination, we calculate *Dual gap* as 100*(CPLEX LB-Dual)/CPLEX LB.

	LP primal gap	LP dual gap	Primal gap	Dual gap
Traditional	46.91%	19.29%	4.47%	3.73%
Random	46.25%	13.74%	3.16%	2.45%
Convex Hull	46.78%	10.25%	2.77%	1.70%
Row Covering	47.00%	9.27%	2.45%	1.28%

Table 10: Primal and dual performance by strategy

We see that performing a partial decomposition yields both a higher-quality primal solution and a stronger dual bound. We also note that, how the partial decomposition is performed has an impact, with the Row Covering strategy yielding the highest quality primal solutions and strongest dual bounds. We attribute the stronger dual bounds produced at the end of the LP phase when using partial decomposition strategies to the

fact that while solving the LP, we dynamically add disaggregate inequalities $x_{ij}^{ks} \leq y_{ij}$, for the scenarios we retain in a cutting plane-type procedure.

Finally, we study in Table 11 the tradeoffs in two statistics related to branch and bound performance with performing a partial decomposition as opposed to a full decomposition. Specifically, we study the strength of the root node bound (*Root node gap*) and the average amount of time spent processing a node in the branch-and-bound tree (*Time per node*). To evaluate the strength of the root node bound (*Root bound*), we calculate the gap between that bound and the lower bound provided by CPLEX after solving the instance for two hours, CPLEX LB, with the formula 100*(CPLEX LB-Root bound)/CPLEX LB.

	$\|S\| = 16$		$\|S\| = 32$		$\|S\| = 64$	
Strategy	Root node gap	Time per node	Root node gap	Time per node	Root node gap	Time per node
Traditional	12.73%	0.09	12.48%	0.18	10.78%	0.45
Random	10.12%	0.20	9.85%	0.36	9.87%	0.64
Convex Hull	6.99%	0.81	8.35%	1.07	8.13%	1.49
Row Covering	7.75%	0.91	9.67%	1.23	8.75%	1.68

Table 11: Branch and bound statistics by strategy

One motivation for performing a partial decomposition was that, with the additional structure in the master problem, CPLEX would be able to generate valid inequalities to strengthen the root node bound. The results indicate that this does happen as the Root node gap is much smaller when performing partial decomposition. We also see that how the partial decomposition is performed impacts the root node gap as well, as it is often much smaller when using the Row Covering or Convex Hull strategies than when randomly choosing the scenarios to retain. However, Table 11 also indicates the drawback associated with performing a partial decomposition; that it takes longer to process each node. We attribute our inability to solve more instances when performing a partial decomposition, as seen in Tables 3a, 3b,6a, and 6b, to this phenomenon. While the increase in time per node may appear as a drawback of performing partial decomposition, we believe it is strongly compensated for by advances in computing and solver power.

5 Conclusions and Future Work

We have proposed the concept of partial decomposition and developed a theory to support it that illustrates how it may be applied to any stochastic integer program with continuous recourse. Such programs are used to model many practical applications such as the one considered in this paper, network design. They are also useful for solving problems with integer recourse as many solution methods for such problems also solve one of its linear relaxations. With an extensive computational study, we have demonstrated the advantages of using a partial decomposition.

Our computational results show that using partial decomposition can greatly reduce the number of optimality and feasibility cuts generated when solving a stochastic program with a Benders-based algorithm and that how the partial decomposition is performed has a significant impact. The theory suggesting how to perform partial decomposition is oriented, however, towards reducing the need for feasibility cuts for the scenario subproblems that are not retained. Hence, one avenue for future work is to develop a theory to apply the strategy that is oriented towards reducing the need for optimality cuts to express the expected recourse cost. Finally, we have focused on static partial decomposition; i.e., the scenario subproblems to retain are chosen in an initialization step of a Benders-based algorithm, which then proceeds to solve the resulting formulation. Hence, another avenue for future work is to consider changing the partial decomposition during the execution of a Benders-based algorithm.

Acknowledgments

Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC) and by the Fonds québécois de la recherche sur la nature et les technologies (FQRNT). This support is gratefully acknowledged.

References

[1] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[2] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[3] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer Series in Operations Research and Financial Engineering. Springer, 2011.

[4] J. R. Birge and F. V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34:384–392, 1988.

[5] C. C. Carøe and J. Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83:451–464, 1998.

[6] G. Codato and M. Fischetti. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766, 2006.

[7] I. Contreras, J.F. Cordeau, and G. Laporte. Benders decomposition for large-scale uncapacitated hub location. *Operations Research*, 59:1477–1490, 2011.

[8] A. M. Costa. A survey on benders decomposition applied to fixed-charge network design problems. *Computers and operations research*, 32(6):1429–1450, 2005.

[9] G. Côté and M. A. Laughton. Large-scale mixed integer programs: Benders-type heuristic. *European Journal of Operational Research*, 16(3):327–333, 1984.

[10] T. G. Crainic, M. Hewitt, and W. Rei. Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. *Computers and Operations Research*, 43:90–99, 2014.

[11] Teodor Gabriel Crainic, Antonio Frangioni, and Bernard Gendron. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, 112(1):73–99, 2001.

[12] Teodor Gabriel Crainic, Xiaorui Fu, Michel Gendreau, Walter Rei, and Stein W Wallace. Progressive hedging-based metaheuristics for stochastic network design. *Networks*, 58(2):114–124, 2011.

[13] R. S. de Camargo, G. de Miranda Jr., and H. P. L. Luna. Benders decomposition for hub location problems with economies of scale. *Transportation Science*, 43(1):86–97, 2009.

[14] M. Fischetti, D. Salvagnin, and A. Zanette. A note on the selection of benders' cuts. *Mathematical Programming*, 124:175–182, 2010.

[15] A. M. Geoffrion. Elements of large-scale mathematical programming part 1: Concepts. *Management Science*, 11:652–675, 1970.

[16] A. M. Geoffrion. Elements of large-scale mathematical programming part 2: Synthesis of algorithms and bibliography. *Management Science*, 11:676–691, 1970.

[17] A. M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10(2):237–260, 1972.

[18] A. M. Geoffrion and G. W. Graves. Multicommodity distribution system design by benders decomposition. *Management Science*, 20(5):822–844, 1974.

[19] K. Holmberg. On using approximations of the benders master problem. *European Journal of Operational Research*, 77(1):111–125, 1994.

[20] J. N. Hooker and G. Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96:33–60, 2003.

[21] W. Klibi and A. Martel. Scenario-based supply chain network risk modeling. *European Journal of Operational Research*, 223:644–658, 2012.

[22] W. Klibi, A. Martel, and A. Guitouni. The design of robust value-creating supply chain networks: A critical review. *European Journal of Operational Research*, 203:283–293, 2010.

[23] T. L. Magnanti and R. T. Wong. Accelerating benders decomposition algorithmic enhancement and model selection criteria. *Operations Research*, 29(3):464–484, 1981.

[24] S. Marsland. *Machine learning: an algorithmic perspective*. Chapman & Hall/CRC, 2009.

[25] D. McDaniel and M. Devine. A modified benders partitioning algorithm for mixed integer programming. *Management Science*, 24:312–319, 1977.

[26] Nemhauser, G.L. and Wolsey, L.A. *Integer and Combinatorial Optimization*. Wiley, New York, NY, 1988.

[27] N. Papadakos. Practical enhancements to the magnanti-wong method. *Operations Research Letters*, 36:444–449, 2008.

[28] C. A. Poojari and J. E. Beasley. Improving benders decomposition using a genetic algorithm. *European Journal of Operational Research*, 199:89–97, 2009.

[29] W. Rei, J.-F. Cordeau, M. Gendreau, and P. Soriano. Accelerating benders decomposition by local branching. *INFORMS Journal on Computing*, 21(2):333–345, 2009.

[30] R. T. Rockafellar and S. Uryasev. The fundamental risk quadrangle in risk management, optimization and statistical estimation. *Surveys in Operations Research and Management Science*, 18:33–53, 2013.

[31] T. J. Van Roy. Cross decomposition for mixed integer programming. *Mathematical Programming*, 25(1):46–63, 1983.

[32] G. K. D. Saharidis and M. G. Ierapetritou. Improving benders decomposition using maximum feasible subsystem (mfs) cut generation strategy. *Computers and Chemical Engineering*, 34:1237–1245, 2010.

[33] G. K. D. Saharidis, M. Minoux, and M. G. Ierapetritou. Accelerating benders method using covering cut bundle generation. *International Transactions in Operational Research*, 17:221–237, 2010.

[34] S. Sen and J. L. Higle. The c^3 theorem and a d^2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, 104:1–20, 2005.

[35] S. Sen and H. D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106:203–223, 2006.

[36] H. D. Sherali and B. J. Lunday. On generating maximal nondominated benders cuts. *Annals of Operations Research*, 210:57–72, 2013.

[37] R. Van Slyke and R. J.-B. Wets. L-shaped programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663, 1969.

[38] G. Zakeri, A. D. Philpott, and D. M. Ryan. Inexact cuts in benders decomposition. *SIAM Journal on Optimization*, 10(3):643–657, 2000.